.NET Frameworkで作る Windowsサーバー

作ればわかるアプリケーションの動作とメカニズム

第7回 ロードバランシング機能付き非同期通信Webサーバーを作る

秋月巌ソリューション事務所
秋月 巌 AKIZUKI, Iwao
http://www.akizuki.co.jp/

前号までのサンプルの問題

前回は複数の接続があったときに、別のスレッドで実行するサーバープログラムの作り方を紹介した。しかし、前回紹介した方法だけでは、新しい接続を順繰りに割り当てるだけなので、負荷のバランスが特定のスレッドに偏る可能性がある。そこで今回は、接続数の少ないスレッドに対して、新しい接続を割り当てるロードバランシング機能を追加した。

その前に、前号までのサンプルのクライアントプログラムに問題があったので、それを解決した。前号までのクライアントプログラムは非同期通信ではなく同期通信ソケットプログラムとして実装していた。同期通信のほうがプログラムコードとして簡単だし、また、非同期サーバーだからといってクライアントも非同期でなければいけないというわけではないことを示したかったからである。しかし、この同期通信クライアントは、正常にシャットダウンができない。アプリケーションウィンドウを閉じても、別スレッドで実行されている受信のためのループが残ってしまうのである。この問題はサーバー側でロードバランスをとるためにクライアントの接続の有無を確認するときに発見された。フォームはすでに閉じているのに、プロセスの中のループが生きているため、クライアントプログラムがいつまでも接続されているように振る舞うのである。そこで、クライアントも非同期通信ソケットプログラムに変更した。

Level
1 2 3 4 5

Technology Tools
- ✓ Visual Basic
- ○ Visual C#
- ○ Visual C++
- ○ SQL Server
- ○ Oracle
- ○ Access
- ○ ASP.NET
- ○ Other:

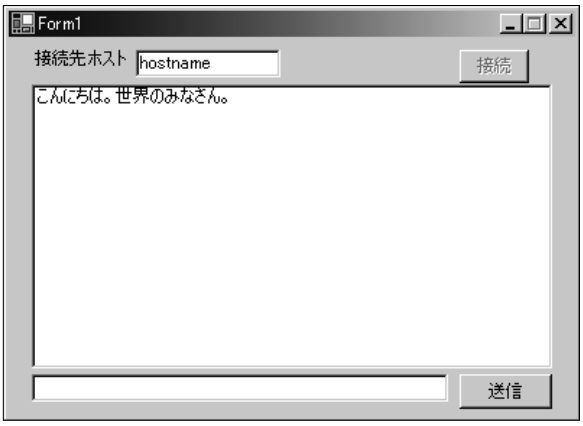
Samples
この記事で取り上げたソースコードおよびサンプルプログラムは、
http://www.shoeisha.com/mag/windev/
からダウンロード可能です。

Webサーバーも非同期通信に

そして、以前作成したWebサーバーも非同期通信複数スレッド、ロードバランス機能付きに修正した。また、このWebサーバーは、Webブラウザとの接続を維持するKeep-Aliveにも対応した。このような修正で、より高い負荷にも対応できるはずだが、実際にどの程度の負荷に耐えられるのかは未知の部分が多い。.NET Frameworkを使用したSocketサーバーというのも、あまり実績はないと思う。

それでも、.NET Frameworkを

図1：サンプル1非同期通信クライアント



使用して実用レベルを目指したWebサーバーのソースコードが提供されることはあまりないと思うので、発表する価値はあると考える。ただ、あくまで技術解説のサンプルなのでエラーハンドラが最低限なのは理解していただきたい。

非同期Socketクライアントプログラム

サンプル1（図1）は今まで同期サーバーとして実装していたSocketクライアントプログラムを非同期通信に変更したもので、外観上の変更はない。また、前号までのサーバーに接続することも可能である。非同期Socketサーバーとの違いは、接続の部分である。リスト1は接続のためのサブプロシージャである。

［接続］ボタンがクリックされると「Button1_Click」サブプロージャが呼び出され「StartClient」サブプロシーシャが別のスレッドで実行される。このプログラムコードは次の行である。

```
HostThread = New Thread(AddressOf StartClient)
```

別スレッドで呼び出された「StartClient」サブプロシーシャの次の行で、接続処理が開始される。

```
SocketClient.BeginConnect(remoteEP, _
 AddressOf ConnectCallback, SocketClient)
```

このSocketクラスのBeginConnectメソッドは、サーバー側では同クラスのBeginAcceptメソッドに対応する。接続が終了するのは、このメソッドの引数で指定されている「ConnectCallback」サブプロシージャが呼び出され、次のようにEndConnectメソッドが呼び出されたときである。

リスト1：サンプル1の接続で使用されるサブプロシージャ

```
Private Sub Button1_Click( _
  ByVal sender As System.Object, _
  ByVal e As System.EventArgs) Handles Button1.Click
    ipHostInfo = Dns.Resolve(TextBox1.Text)
    ipAddress = ipHostInfo.AddressList(0)
    Button1.Enabled = False
    Button2.Enabled = True

    HostThread = New Thread(AddressOf StartClient)
    HostThread.Start()
    Thread.Sleep(1000)
End Sub

Private Sub StartClient()
    Dim remoteEP As New IPEndPoint(ipAddress, 11000)
    SocketClient = _
      New Socket(AddressFamily.InterNetwork, _
      SocketType.Stream, ProtocolType.Tcp)

    SocketClient.BeginConnect(remoteEP, _
      AddressOf ConnectCallback, SocketClient)
    connectDone.WaitOne()

    Receive(SocketClient)
End Sub

Private Sub ConnectCallback(ByVal ar As IAsyncResult)
    Dim client As Socket = CType(ar.AsyncState, Socket)

    client.EndConnect(ar)
    connectDone.Set()
    receiveDone.Set()
End Sub
```

```
client.EndConnect(ar)
```

接続が行なわれた後でのデータ通信の処理に関しては、サーバーとクライアントで差がないので解説は省略する。

ロードバランシング機能付き非同期通信サーバー

前号のサーバーに、ロードバランス機能を追加したのが、このサンプル2（図2）である。クライアントからの接続があったときに、もっとも少ない接続しか扱っていないスレッドに接続を割り当てる。この機能はクライア